AFRL-RI-RS-TR-2013-069

AUTOMATED FEATURE SELECTION FOR EXPERIENCE-BASED ADAPTIVE RE-PLANNING

STATE UNIVERSITY AT BINGHAMTON

MARCH 2013

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2013-069 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /

KURT LACHEVET
Work Unit Manager

/ S /

JULIE BRICHACEK
Chief, Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MARCH 2013	FINAL TECHNICAL REPORT	MAR 2012 – SEP 2012

4. TITLE AND SUBTITLE

AUTOMATED FEATURE SELECTION FOR EXPERIENCE-BASED ADAPTIVE RE-PLANNING

5a. CONTRACT NUMBER
FA8750-12-2-0130

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)
Lei Yu

5d. PROJECT NUMBER
S2EB

5e. TASK NUMBER
AF

5f. WORK UNIT NUMBER
SE

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Research Foundation Of State University
SUNY At Binghamton
85 Murray Hill Rd
Binghamton, NY 13902

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RISC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2013-069

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2013-1037
Date Cleared:5 MAR 2013

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project proposes innovative methodologies of automatic feature selection for Experience-Based Adaptive Replanning (EBAR). EBAR is an extension of the Distributed Episodic Exploratory Planning (DEEP) in-house project conducted at the Air Force Research Lab. This project pursues two complementary research objectives: (i) developing efficient feature selection algorithms for case based planning (CBP), and (ii) evaluating and demonstrating the effectiveness of feature selection for CBP. In this project, we have successfully developed two major types of feature selection methods for CBP, wrapper and filter, which differ mainly in how they evaluate the quality of a feature subset. We have also developed an efficient result validation procedure and demonstrated the efficiency and efficacy of the proposed feature selection methods based on the StarCraft domain.

15. SUBJECT TERMS
Case Based Reasoning, Feature Selection, Classification, Clustering, Similarity Measure

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**KURT LACHEVET**
U	U	U	UU	32	19b. TELEPONE NUMBER (*Include area code)* **N/A**

Table of Contents

List of Figures

List of Tables

1. Executive Summary

This project pursues two complementary research objectives: (i) developing efficient feature selection algorithms for case based planning (CBP), and (ii) evaluating and demonstrating the effectiveness of feature selection for CBP.

A key obstacle in developing planning and adaptive replanning methods for real-world problems is the so-called "curse of dimensionality" - the state space explosively grows as the number of state variables (features) increases, making planning/replanning methods prohibitively expensive to compute. This project aims to improve the scalability of case based planning/replanning in domains with high-dimensional state spaces by automatic feature selection.

This project proposes innovative methodologies of automatic feature selection for Experience-Based Adaptive Replanning (EBAR). EBAR is an extension of the Distributed Episodic Exploratory Planning (DEEP) in-house project conducted at the Air Force Research Lab. The basic approach for EBAR is CBP. In CBP, past planning experiences are stored in a case base where each case describes a situation (the state of the environment) and the plan for that situation. A candidate plan for each new situation is decided by retrieving a case of the most similar situation from the case base. Given potentially hundreds of features which can be used to describe the state of the environment, automatically selecting a small set of features that are sufficient for efficient and effective CBP is one of the key challenges facing the DEEP team. Currently, there is no existing solution to this challenge.

In this project, we have developed two major types of feature selection methods for CBP, wrapper and filter, which differ mainly in how they evaluate the quality of a feature subset. A wrapper feature selection method evaluates the quality of a feature subset based on how similar the plan of a case to the plan of its nearest neighbor (the case with the most similar situation) measured based on the subset of state features. In contrast to a wrapper method, a filter method does not decide feature relevance based on nearest neighbor search, but based on how well a subset of state features distinguishes cases of dissimilar plans. For both wrapper and filter methods, the key innovation in feature subset evaluation lies in a general similarity (or distance) measure for the plan portion of cases in a case base. We have developed novel measures for plan similarity and feature selection methods based on these measures. We have developed an efficient result validation procedure and demonstrated the efficiency and efficacy of the proposed methods based on the StarCraft domain.

2. Introduction

As reported in the recent Air Force Chief Scientist "Technology Horizons" [3], natural human capacities are becoming increasingly mismatched to the enormous data volumes, processing capabilities, and decision speeds that technologies either offer or demand. Two key areas in which significant advances are possible in the next decade are: (i) increased use of autonomy and autonomous systems, and (ii) augmentation of human performance. Both areas can achieve capability increases and cost savings via increased manpower efficiencies and reduced manpower needs. Mission planning is one of the domains which will greatly benefit from such capability increases and cost savings.

The complex and real-time nature of these problems requires commanders to make timely decisions and makes a complete search of the domain for planning impractical. To move towards autonomous adaptive replanning, we require methods of determining that a plan has deviated from its expected performance, and performing adaptive replanning as necessary to reduce the differential caused by these deviations.

A key obstacle in developing planning and adaptive replanning methods for large-scale real-world problems is the so-called "curse of dimensionality" - the state space explosively grows as the number of state variables (features) increases, making planning/replanning methods prohibitively expensive to compute. This project aims to improve the scalability of case based planning/replanning in domains with high-dimensional state spaces by automatic feature selection. *To reach this goal, the project pursues two complementary research objectives: (i) developing efficient feature selection algorithms for case based planning (CBP), and (ii) evaluating and demonstrating the effectiveness of feature selection for CBP.*

There are two main challenges in feature selection for CBP: how to evaluate the relevance of a feature subset without supervised class label and how to validate the result of a feature selection algorithm for CBP. In this project, we address both challenges by solutions which are centered on an innovative way of measuring case similarity. The basic idea is to treat the plan portion of a case as its supervised label, and represent each label in the form of a sparse matrix, enabling the design of a general loss function to measure the similarity (or distance) between two cases based on their plan portions. Given the general similarity measure, we have developed two major types of feature selection methods for CBP, wrapper and filter, which differ mainly in how they evaluate the quality of a feature subset. A wrapper feature selection method evaluates the quality of a feature subset based on how similar the plan of a case to the plan of its nearest neighbor (the case with the most similar situation) measured based on the subset of state features. In contrast to a wrapper method, a filter method does not decide feature relevance based on nearest neighbor search, but based on how well a subset of state features distinguishes cases of dissimilar plans.

Based on the general similarity measure, we have also developed an efficient result validation procedure based on N-fold cross validation and demonstrated the efficiency and efficacy of the proposed methods based on the StarCraft domain.

3. Methods, Assumptions and Procedures

3.1. Background and Related Work

3.1.1. Case Based Planning

Planning can be viewed as a sequential decision making process. Formally, such processes can be modeled as a Markov Decision Process (MDP) [10]. A MDP is represented as a 4-tuple (*S,A,T,R*), where: *S* is a set of states, *A* is a set of actions, *T* is a transition function, and *R* is the reward function. In a factored MDP, the set of states *S* is determined by a set of state variables (features). In the planning system, an existing intelligence surveillance and reconnaissance/ assessment service provides information to build a representation of the current state of the executing plan. By understanding the state variables, the CBP system can retrieve relevant actions to be executed. The case base represents individual plans as Markov Chains, containing sequences of states and actions which occurred to reach an objective. Because this is a model-free approach, we do not have complete models of the entire state space and therefore the probability of an action a moving from the current state *s* to the desired state s' is not 1. Similarly, we are also not undertaking the modeling of the transitional probabilities from state to state. CBP allows uncertainty management by drawing upon past experience to choose an action set based on the current state variables. Therefore, when the state variables are defined well enough, it can then be used to retrieve Markov Chains from the experience base to plan by providing sequences of states and actions to solve current objectives.

A central challenge to the EBAR project is to automatically construct and identify optimal or near-optimal feature sets for case retrieval and plan monitoring. Features are the sensors and variables that describe the current state of the environment. A concise and complete feature set is critical for managing the size of a search space and ensuring a decision maker, human or synthetic, has enough information to make an appropriate decision. In practice, features are typically manually selected and derived by subject matter experts. Manually selecting features is expensive, error prone, and inflexible. An automated method for feature selection would be much more robust, less costly, and more capable in dynamic domains.

3.1.2. StarCraft Simulation Platform

In the DEEP research platform [2], the simulation is the StarCraft real-time strategy game. StarCraft is a popular military science fiction force-on-force real-time strategy game developed by Blizzard Entertainment which was originally released in 1998. Game play includes force against force engagements where a red and blue force build an economy, base, and depending on choices made develop certain capabilities to employ against their enemy. High level aspects of the game are the development of the economy vs. the development of the army. If one builds too heavy of an economy they are susceptible to attacks from an opponent's army. However, the benefit of building a strong economy is to improve technology and build a more formidable army later in the game. There are many other important aspects to the game which help a player achieve victory, such as effective scouting of the opponent, setting up proper defenses, and countering the opponent's capabilities, etc. This domain is chosen by the DEEP team as the

simulation platform for the following main reasons.

- Large case base: one of the most important requirements of a domain for experience-based reasoning is the ability to obtain a large corpus of cases to use to reason over. StarCraft has the ability to save a "replay" of a game and there are a huge number of data stores of these replays readily available.

- Complexity: although the simulation engine is deterministic, because of the large action space it is unlikely to see the same outcomes given a set of actions in a game.

- Active research domain: this domain has been used in several United States Navy sponsored contracts and is also an active research domain by a number of academic institutions.

- Abstract Command and Control domain: this domain allows planning and coordination of multiple entities throughout the full spectrum from tactical to strategic military operations.

- Other benefits: such as asymmetric forces (different races, planning and coordinating differing objectives), uncertainty, quantitative metrics, real-time constraints, rich feature set, and interface for human interaction.

The DEEP team is currently focusing on the strategic level of planning as they believe that is where experiential reasoning is well suited. There are components of the application programmers interface (API) that allow the use of "managers" to handle the tactical level tasks of the game. These managers include:

- Resource manager - this handles the gathering of resources in the game including the production of these resource gatherers.

- Building manager - this alleviates the need of spending time writing algorithms to deal with coordinate placement on the map.

- Attack manager(s) - they decide tactical-level actions and behaviors to apply to specific units.

Figure 1 shows a simplistic version of the StarCraft planning architecture and how the DEEP system interacts with it. The central component is the Blackboard which is a GOTS tool (see [2] for more information) to support easy, rapid research and testing. The blue objects are the information passed between the StarCraft APIs and DEEP components. The "situation" object is the set of information populated by the StarCraft APIs. This contains information such as map details, unit details, building details, red information, blue locations, etc. The features being tracked is currently under development. The current implementation of a plan in this domain consists of capabilities you wish to develop at certain stages of the game. The actions to develop these capabilities include: types of buildings and structures to build, upgrades to research, and number and compositions of units to create in order to achieve certain capabilities.

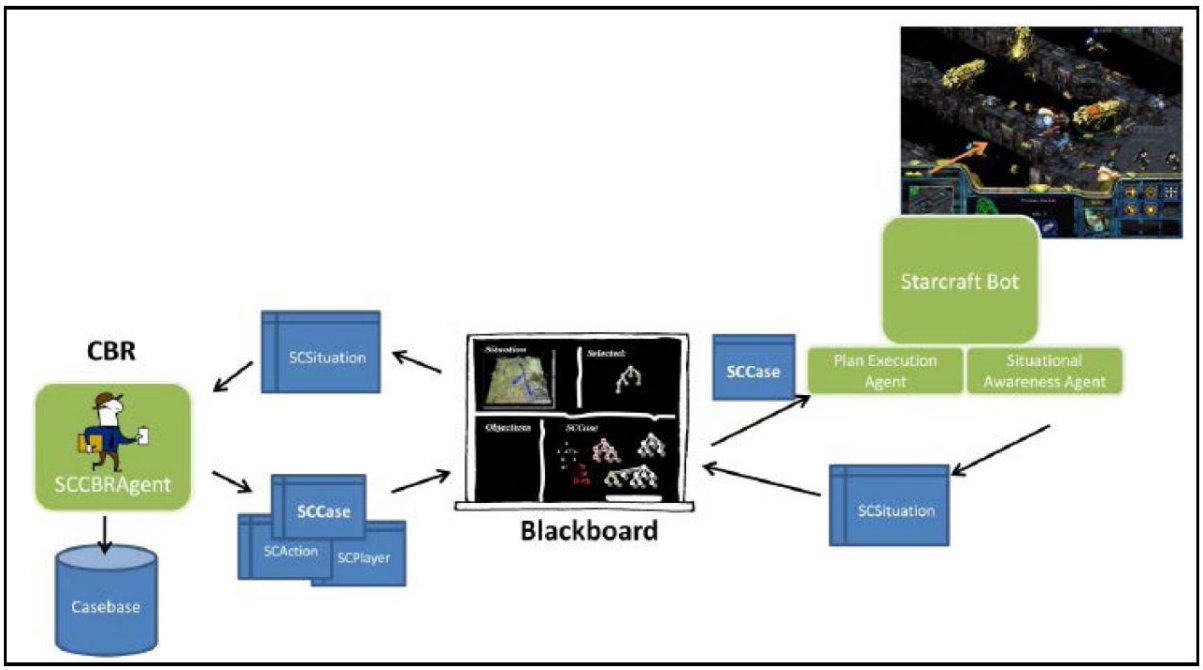

Figure 1. StarCraft Planning Architecture (Provided by the DEEP Team)

3.1.3. StarCraft Case Base

Once the DEEP team settled on StarCraft as a domain of simulation, a case base had to be developed to support the CBP process. These steps included designing case structure, selecting features, formal logic for a StarCraft plan, and automation logic to convert StarCraft replays into cases.

As described in the previous section, one benefit to utilizing StarCraft as a simulation domain is the large amount of publicly available replays. Using built in game functionality to save replays, it is easy to extend the case base over time. A replay is a saved game that can be viewed later on or even executed again to monitor what happened. Combining an open source API for parsing replays with a developed programmatic option to automate reading the replays and parsing out the required information, the DEEP team was able to quickly generate thousands of cases.

Before creating the case base, the DEEP team conducted internal discussions using empirical evidence of game play to support a selection of features that would support initial case construction. It was also decided that the cases to be used would be one player versus one player matches, so no games with more than two players are implemented in the platform. The importance of what features to use was an iterative process. As the project continued the team reviewed the work of other teams that were developing their own AI systems, such as bots, to be used at the StarCraft AI conference such as the Berkeley Overmind Project (http://overmind.cs.berkeley.edu) to see what other features were being created by the AI community for possible inclusion into the DEEP system.

Table 1. Features for StarCraft Case Base

Game Features	Player Features
Unique Identifier	Player name
Replay name	Player race
Map name	Count of buildings built
Winning player name	Count of units built
Winning player final state lattice	Count of workers built
Actions per minute of game	Expansions and their game frame
	Final player state lattice
	Vector of player states by time step
	Unit type and production count at each time step

Table 1 lists the current set of features used for describing a case in the existing StarCraft case base. A time step in each case is two thousand game frames, so for some of the features, the feature values were measured over every two thousand frames. The frames are how the game steps through the simulation and they run at 24 frames per second. The state lattice is based on the build tree (or tech tree expansion) of StarCraft (Broodwar Expansion Pack). Specific buildings enable new features and capacities like: building new types of combat units and buildings, researching new capabilities, etc. The SCReplay (www.cs-replay.com), an open-source java based project was utilized to read contents of saved StarCraft games. This approach enabled the DEEP team to effectively reduce the complexity of representing an entire game state through segmentation.

The DEEP team feels feature selection for this project is not complete and requires future work to support better plan generation, more specific player information, more specific mapping information, among others. Most of the information in a case is used by retrieval to obtain the best match to a given situation; however there is part of the case structure that is utilized later on by the plan execution agent to implement the plan. All the information under the units at each time step describe what was built and at what time. This is the build order the player used in the game and consequently, is the action list the plan execution agent will use to construct buildings and build units. The data for player expansions is also used to allow the plan execution agent know when to build an expansion. Figure 2 provides an example of feature data and the XML case base for StarCraft.

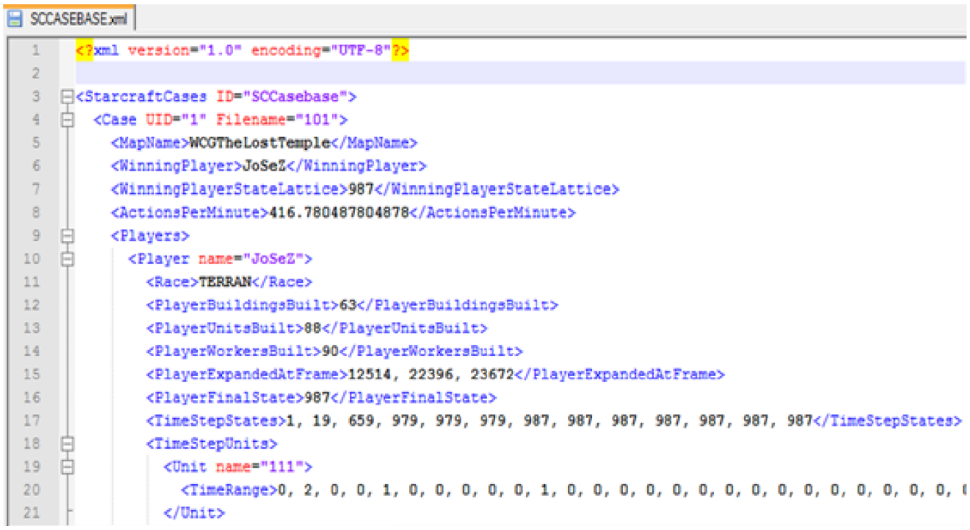

Figure 2. An Example of Feature Data and Case Base for StarCraft (Provided by the DEEP Team)

3.1.4. Feature Selection

Feature selection, the problem of automatically selecting a subset of the original features for a learning task, has been an active field of research for many years (reviewed in [7,12,15]). Various feature selection techniques can be categorized into supervised [1,7-9] and unsupervised [5,6,20] depending on the availability of class labels. Let $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be a training set of n labeled instances, where $\mathbf{x} \in \Re^d$, defined by d features $X_1, \ldots, X_d$, and y is the value of the class variable Y . The goal of supervised feature selection is to select a minimum subset $S^* \subseteq X$ such that $p(Y|S^*) = p(Y|X)$, where $p(Y|S^*)$ and $p(Y|X)$ represent the conditional distributions of class values given feature values in S^* and X, respectively [9]. Due to the exponential search space w.r.t. the dimensionality and the imprecise approximation to $p(X,Y)$ by a training set, it is neither feasible nor appropriate to search for S^*. In practice, feature selection algorithms resort to a combination of practical evaluation measures and efficient search strategies.

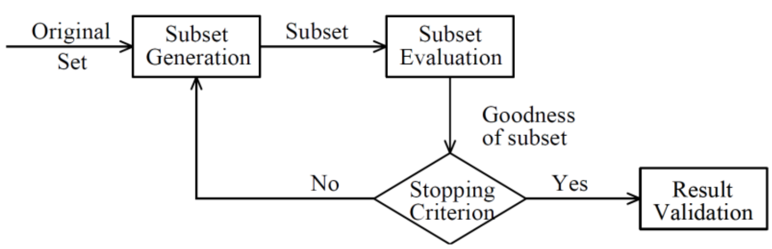

Figure 3. A General Framework of Feature Selection

Figure 3 shows the general framework of a feature selection process. Various feature selection algorithms differ mainly in the subset generation (search strategy) and subset evaluation components. A number of search strategies can be used for generating a candidate feature subset for evaluation, such as complete search, sequential search, or genetic programming. Various

algorithms can also be broadly grouped into wrapper [8, 13] and filter [14, 21] methods, depending on the evaluation measures. A wrapper method requires a predetermined learning algorithm and uses the performance of the learning algorithm applied on the selected subset to determine which features are selected. It usually gives superior performance as it finds features better suited to the predetermined learning algorithm than filter methods, but it also tends to be more computationally expensive. A filter method tries to evaluate the goodness of a feature or feature subset by exploiting the intrinsic characteristics of the training data without involving any learning algorithm. Some popular filter measures include distance, information gain, dependency, and consistency measures [12]. A stopping criterion determines when the feature selection process should stop - usually, when a specified number of features is reached, or subsequent addition (or deletion) of any feature does not produce a better subset. A straight-forward way for result validation is to directly measure the feature selection result based on prior knowledge about the set of relevant features of the data. When prior knowledge is not available as in most real-world applications, we can monitor the change of learning performance with the change of features. For example, given a finally selected feature subset, we can compare the generalization errors on an independent test set made by the two classifiers learned on the training set using the full set of features and the selected subset of features.

Although feature selection has been extensively studied in supervised learning, it has received little attention in the areas of case-based reasoning and planning. To the best of our knowledge, Mishra et al.'s work on situation assessment for plan retrieval [16] is the only study that addresses feature selection for case-based planning. A situation is a high-level representation of the state of the world, and situation is predicted using a decision tree based situation-classification model based on "raw" state features. Situation prediction is further used for the selection of relevant "knowledge intensive" features which are derived from the basic representation of the environment. The similarity of cases is computed based on the selected relevant features. The feature selection performed here is "knowledge" centered and closely related to state space reduction by subtask decomposition [4]. The feature selection methods proposed in this proposal are "data" centered, which directly work on the case base to identify relevant features without relying on the knowledge of subtask decomposition of the problem. These methods can be integrated with the knowledge-centered approach in the future.

3.2. Proposed Methodologies

In this section, we present our proposed methods for automatic feature selection for case-based planning (CBP), assuming a case base is already constructed according to a set of predefined features by subject matter experts (as illustrated in Section 3.1.3). In Section 3.2.1, we propose a novel similarity measure for the plan portions of cases, which serves as a common basis for feature quality evaluation in the feature selection methods developed in Section 3.2.2 and the result validation metrics in Section 3.2.3.

3.2.1. General Similarity Measure for Plans

Like feature selection for supervised learning, feature selection for CBP also needs to address two key issues: (1) efficient search of the exponential space of possible feature subsets, and (2)

efficient and effective evaluation of the relevance of a feature or feature subset. Various search strategies developed for supervised feature selection can be adopted for CBP. However, the feature evaluation measures used in wrapper and filter feature selection methods for supervised learning are not readily applicable to the CBP setting due to the lack of traditional class labels for cases. Unlike in supervised learning where the label of each instance is a symbolic or numeric value of a random variable, in CBP, how to define the label for each case and measure the similarity among different labels is an open research issue.

In our approach of feature selection for CBP, we treat the plan portion (the sequence of actions) in each case as the case label, and apply a general similarity measure for measuring the match between the two label values of a retrieved case and a test case. Specifically, in the current implementation of the case base, the plan portion of each case can be viewed as a sparse matrix $\mathbf{P}$, where each column vector $\mathbf{p}_t$ represents a snapshot of actions taken at a given time step t, and each row vector $\mathbf{p}'_r$ represents the execution sequence of a given action a across all time steps, and each value $p_{r;t}$ in the matrix represents how many times action a is executed at time step t. In the case base example in XML format (Figure 2), the row <TimeRange> shows a truncated (from 70 time steps) sparse vector encoding the number of units built for Unit named "111". For efficient storage purposes, the XML case base only stores the row vectors for actions that are taken at least once for each case. The sparse matrix representation eases our conceptual discussion of the similarity measure, and the conversion from XML format is trivial.

Given the above matrix representation, we can define a general loss function $L(\mathbf{P}, \mathbf{P})$ which measures the cost of retrieving plan $\hat{\mathbf{P}}$ when the true plan of case $\mathbf{x}$ is $\mathbf{P}$ as:

$$L(\mathbf{P}, \mathbf{P}) = \frac{1}{T}\sum_{t=1}^{T} d(\mathbf{p}_t, \mathbf{p}_t) w_t \quad , \tag{1}$$

where T is the total number of recorded time steps, $d(\mathbf{p}_t, \mathbf{p}_t)$ is a distance function between two sparse vectors, and w_t is the weight assigned to each time step t. The general loss function can have various forms depending on the choices of the distance function and the weight distribution. For example, we can use the well-known Cosine distance and l_1 and l_2 norms between two vectors for the distance function. For the weight distribution, besides equal weighting, we can assign higher weight for earlier time steps assuming differences in actions that occurred earlier in two plans will have bigger impacts on the outcome. Under this general form of loss function, the labels of two cases are considered more similar if the loss function returns smaller values.

3.2.2. Feature Selection Methods for CBP

Based on the general similarity measure proposed earlier, we can develop both wrapper and filter types of feature selection algorithms for CBP. The basic idea of wrapper feature selection is to repeatedly generate candidate feature subsets following a search strategy and rely on the performance of a predefined learner/planner to evaluate the goodness of a feature subset. In CBP, to determine the plan of a new case (representing a new situation), the nearest neighbor (the case with the most similar situation to the new case) from the case base is retrieved, and the plan of the nearest neighbor is recommended as the candidate plan for the new case. Ideally, given the

optimal set of features to represent all possible cases of a given task and a sufficiently large case base with known optimal plans, the nearest neighbor of any new case would recommend a plan that is as close as possible (if not the same) to the optimal plan for the new case. This is the essence of why CBP works. Obviously, the nearest neighbor returned is sensitive to the set of state features used in calculating the similarity (or distance) between two cases. Therefore, we can develop a wrapper feature selection method which evaluates the quality of a feature subset based on how similar the plan of a query case to the plan of its nearest neighbor (determined based on the subset of state features) for all cases in the case base. Intuitively, the higher the overall similarity, the better the feature subset is. We next formally introduce the feature quality measure for wrapper feature selection in the CBP setting by extending the idea of wrapper feature selection based on nearest neighbor learning [11, 17].

Given a training set $D = \{(\mathbf{x}_1,y_1),\ldots, (\mathbf{x}_n,y_n)\}$, where $\mathbf{x} \in \mathcal{R}^d$ and y is the value of the class variable Y, and a test instance $\mathbf{x}$ with true class label y, a nearest neighbor classifier $y\hat{} = f(\mathbf{x})$ decides the label of $\mathbf{x}$ by $\mathbf{x}$'s nearest neighbor $\mathbf{x}^N_i$ in D (i.e., $y\hat{} = y_i^N$). A loss function $L(y,y\hat{})$ measures the cost of predicting $y\hat{}$ when the true label of $\mathbf{x}$ is y. For numerical labels (as in regression), commonly used loss functions are squared loss ($L(y,y\hat{}) = (y - y\hat{})^2$) or absolute loss ($L(y,y\hat{}) = |y - y\hat{}|$). For categorical labels (as in classification), zero-one loss ($L(y,y\hat{}) = 0$ if $y = y\hat{}$, $L(y,y\hat{}) = 1$ otherwise) is typically used. Given a feature set S, the quality of S for nearest neighbor learning is measured by the aggregated leave-one-out loss of the training set based on the feature space defined by S:

$$\mathbf{Q}(S) = \frac{\mathbf{1}}{n}\sum_{i=1}^{n} L(y_i.y_j^N) \quad , \tag{2}$$

where y_i and y_j^N are the labels of $\mathbf{x}_i$ and its nearest neighbor $\mathbf{x}_j$ ($i \neq j$) in D according to the subset of dimensions in S. The goal of feature selection for nearest neighbor learning is to select a subset S such that $\mathbf{Q}(S)$ is minimized. A wrapper method can be developed by coupling an efficient search method such as sequential forward or backward search with the evaluation measure in Equation (2).

Given the general loss function in Equation (1), we can extend Equation (2) to the following feature quality measure for CBP:

$$\mathbf{Q}(S) = \frac{\mathbf{1}}{n}\sum_{i=1}^{n} L(\mathbf{P}_i.\mathbf{P}_j^N) \quad , \tag{3}$$

where $\mathbf{P}_i$ and $\mathbf{P}^N_j$ are the labels of case $\mathbf{x}_i$ and its nearest neighbor $\mathbf{x}_j$ ($i \neq j$) in the training case base retrieved based on S. We can combine this measure with sequential backward search or sequential forward search to come up with different wrapper feature selection methods for CBP.

In contrast to a wrapper method, a filter method for CBP does not rely on the nearest-neighbor based planner in deciding feature relevance. A filter method decides the relevance of a feature subset based on how well a subset of state features distinguishes cases of dissimilar plans. Like wrapper methods, the key of feature subset evaluation for filter methods is also the similarity (or distance) measure for the plan portions of cases in a case base. Here we propose two approaches

to apply the general loss function in Equation (1) to build filter methods.

The first approach is to directly use the general loss function in Equation (1) with existing filter methods built for *regression* problems such as RReliefF [19]. The basic idea of RReliefF is to weight and rank each feature based on the following heuristic. Over all pairs of instances, the weight of a feature is higher if the distance between feature values of a pair of instances is more positively correlated with the distance of their class labels, and the weight is lower if the correlation is more negative. For regression problems, the distance of class labels is straightforwardly measured by squared or absolute distance between two numerical values. For CBP, Equation (1) enables us to measure the distance between the labels of two cases.

The other approach is to indirectly use Equation (2) with existing filter methods built for *classification* problems such as FCBF [23] and mRMR [18]. The basic idea of these methods is to measure the relevance and redundancy of features based on mutual information measure. In order to adopt these methods, we need to have a set of categorical labels for labeling each of the cases. This leads to an interesting task - clustering all the cases in a case base into a number of distinct groups based on the plan portion of each case and the distance measure in Equation (1). Methods like hierarchical clustering and spectral clustering which only require a pairwise distance matrix for all cases are suitable for this task. After the clustering preprocessing step, existing filter methods for classification can be applied on the transformed case base with categorical class labels.

Compared with the direct approach, the indirect approach entails the cost of clustering and validation of clustering results. Nevertheless, this cost only occurs once during a preprocessing phase. Besides the benefit of enabling efficient filter feature selection, clustering cases based on plan portions may reveal interesting patterns that correspond to high-level strategic plans and shed light on how to improve the case base construction.

3.2.3. Result Validation Metrics of Feature Selection for CBP

We now address how to validate the result of feature selection for CBP. Following the basic idea of result validation in supervised feature selection, given a finally selected feature subset, we can compare the performance of the case base on recommending plans for an independent test set based on the full set of features and the selected subset of features. The current practice of the DEEP team resorts to a handful of predetermined test cases (known game situations with unknown plans) in evaluating the performance of CBP based on the full set of manually determined state features. Given a test case, the quality of the retrieved nearest neighbor case can be determined by actually executing the plan of the retrieved case and monitoring the outcome of the plan (winning or losing to the opponent player). Although this evaluation practice has demonstrated the promise of CBP for the StarCraft domain, it has three main limitations for evaluating the performance of feature selection for CBP. First, real-time execution of the plan of a retrieved case is very time consuming, and therefore, is inefficient for repeatedly evaluating a number of different feature subsets produced on the same case base by various proposed feature selection algorithms or under different parameter settings of the same algorithm. Second, the outcome of the plan execution does not always reflect the true merit of a strategic plan conveyed

by a retrieved case, and therefore, is not reliable for validating the feature subset used for CBP. As mentioned earlier, the execution of the plan relies on an attack manager that provides tactical-level actions and behaviors to apply to specific units. The current heuristics adopted by the DEEP team for the attack manager are not always optimal for making the best use of the strategic plan of a retrieved case. Third, due to the small size, the current set of test cases may not be representative enough for various possible situations.

To address the above challenges of result validation, we need a practical validation procedure which does not require real-time execution of a retrieved plan. We adapt the widely used procedure of N-fold cross validation (CV) in supervised learning, and implement a CV procedure for CBP. Given a large case base and a feature selection algorithm, the CV procedure randomly partitions the entire case base into N equally sized folds, and repeatedly assigns one of the N folds as an independent test set and the remaining $N - 1$ folds together as a training set. For each training and test assignment, the feature selection algorithm is only applied to the training set to produce a finally selected subset. The effectiveness of the feature selection algorithm is measured based on the overall loss of every case in each of the N test sets compared to its nearest neighbor retrieved from the corresponding training set based on the selected feature subset. The same loss function defined in Equation (1) is used here to measure the loss of a test case to its nearest neighbor. It is worth pointing out that in a wrapper method for CBP, Equation (1) is used to measure the quality of a candidate feature subset based on the "leave-one-out" loss of the training set (as in Equation (3)), while in result validation of any feature selection algorithm, Equation (1) is used to measure the effectiveness of the algorithm based on the CV loss of the entire case base. The proposed CV procedure alleviates the problem of a small hand-picked test set, and the use of the general loss function in the CV procedure avoids the burden of real-time execution of retrieved plans.

Alternatively, we can also use prior knowledge of feature relevance for result validation. Although complete knowledge of the relevance of all possible features for the StarCraft domain is unavailable, the DEEP team does have knowledge of the relevance of a handful of manually selected features for the current case base. We can manually add irrelevant and/or noise features to the case base, and expect the proposed feature selection algorithms to successfully eliminate these added features from the selected feature subset.

4. Results and Discussion

4.1. Case Similarity Analysis

As described in the discussion of the general seminary measure, each unit-type has associated with it a weight that indicates the impact of that unit on the case. In all experiments we uniformly set all weights to one. For experiments involving the action plan matrix, each column vector (unit type) was scaled using min-max normalization to set the matrix values between 0 and 1. Additionally, to measure the distance between two vectors we experimented with cosine, l_1, and l_2 measures. The results reported below use the l_1 measure unless otherwise noted, which was found to slightly increase the performance of the algorithms under study. The case base used for all experiments contained 60 cases from a variety of Star Craft replays.

All feature selection algorithms developed in this project rely on the proposed general similarity measure to evaluate the match between two cases. Here we examine some performance characteristics of the similarity measure to validate its use in other areas of the study. Table 2 reports the average leave one out cross validation (LOOCV) performance when measuring the similarity between a query and its nearest and farthest neighbors, with respect to the similarity measure, along with the average pairwise similarity between a query and all other cases in the case base. To clarify, the nearest neighbor is the case that has the most similar action plan matrix (as measured by the similarity measure) as the query case, and the farthest neighbor has the most dissimilar action plan matrix.

Table 2. LOOCV Performance of the Proposed Similarity Measure

Nearest Neighbor	Farthest Neighbor	Average Neighbor
0.51 ± 0.10	0.05 ± 0.03	0.28 ± 0.07

Since the action plan matrices are normalized and unit weights are used on all columns, the values of the similarity measure are bounded between 1.0 (most similar) and 0.0 (least similar) in all experiments. This shows that the nearest neighbor between a query case and all others in the case base scores around midway in the possible range of the loss function, which is somewhat lower than could be expected, and possible explanations for this are given in the following discussion section. The furthest neighbor is near 0.0, while the average similarity was between the two, as expected. Further experiments show that the similarity between the plan matrix of a case and itself, $L(\mathbf{P}_i, \mathbf{P}_i) = 1.0$, and that the measure is reflexive: $L(\mathbf{P}_i, \mathbf{P}_j) = L(\mathbf{P}_j, \mathbf{P}_i)$ for any cases i and j in the case base. These results show that the proposed measure is able to separate similar and dissimilar cases from one another, and is reasonable to use as a basis in the later feature selection experiments.

4.2. Wrapper Feature Selection

Here we study the performance of the wrapper based feature selection algorithm described in Section 3.2.2. The goal of this experiment is to determine the best subset of the original seven indexing features to use to index the case base. The "best" subset of features is the one in which the nearest neighbor to a query case when measured in that subspace gives the highest action plan similarity value. In this context, nearest neighbor distance between cases is measured with respect to the subset of seven features currently used to index the case base, so each case is represented by a k-component vector, where $1 \leq k \leq 7$. Due to the categorical nature of the features, normal distance measures (i.e. Euclidean and Manhattan, etc.) are not applicable. Instead, distance is calculated as:

$$dist(\mathbf{v}, \mathbf{v}') = \frac{1}{k} \sum_{i=1}^{k} I(\mathbf{v}_i, \mathbf{v}_i'),$$

where $I(\cdot,\cdot)$ is the indicator function returning 0 if the arguments are equal, 1 otherwise, and $\mathbf{v}_i$ denotes the i^{th} component of vector. The wrapper experiment implemented here performs a sequential forward subset search of the set of seven features used to index the case base. For a given feature subset, each feature outside of the subset (each candidate feature) is added to the subset in turn, and the LOOCV similarity of the plan matrices is measured. The feature that results in the highest LOOCV similarity is added to the subset, and the search continues through the remaining candidate features. This process terminates when no candidate feature improves the LOOCV similarity above the value measured on the current subset. To begin the search, each feature is evaluated individually. Table 3 reports the average value and standard deviation on the LOOCV for the best feature subset found by the wrapper process and the LOOCV similarity when the full feature subset is used.

Table 3. LOOCV Similarity Based on the Full Feature Set and Selected Subset by the Wrapper

Full Feature Set	Wrapper Selected Feature Subset
0.38 ± 0.12	0.42 ± 0.14

As we can see from this experiment, finding the nearest neighbor to a query case in the subspace of the selected feature subset identified by the wrapper algorithm outperforms the nearest neighbor found in the full set of features. The feature subset identified by the wrapper consists of three features out of the possible seven (blue race, red race, and red state), and is also best performing subset out of *any combination* of the seven given features, as found by doing an exhaustive search over the 127 possible feature subsets (the power set excluding the empty subset).

The results of Table 3 are also consistent with the results shown in Table 2. When using either the full set or the subset identified by the wrapper we see that the similarity between a query and its nearest neighbor is greater than that of itself to all other cases (the Average Neighbor of Table 2). We do note, however, that the similarity of the nearest neighbor in either scenario of Table 3 is significantly less than the most similar case to the query that exists in the case base, as evidenced in the first column of Table 2. This finding indicates that there is more information in the action plan matrix than is currently being captured in the feature subset, at least from a nearest neighbor query response perspective. These findings justify the use of feature selection for case based retrieval, but we do note that performing wrapper feature selection is computationally expensive since it involves doing many nearest neighbor comparisons during the subset search. In the following section, we consider filter feature selection methods to alleviate this cost.

4.3. Filter Feature Selection

The results in this section consider selecting a relevant subset of index features *without* resorting to measuring the query similarity performance to judge the goodness of a selected subset. In

addition to the 7 known indexing features, we added irrelevant features to the case base. The purpose of such an experiment is to verify the proposed feature selection algorithms are capable of selecting the most relevant features when a large set of (potentially irrelevant) features is available from the case base. In our experiments, each additional feature is a uniform random variable over 10 categorical values, since the relevant features are also categorical over a typically small number of possible values. In addition to comparing wrapper feature selection against no feature selection, we also developed a Relief-based filter feature selection algorithm and present the results in Figure 4. For these experiments, we adopted a 10-fold cross validation procedure, where the data is partitioned into 10 equal sized groups, 9 of which are used to train the feature selection algorithms, and each case in the holdout fold is used a test query; the similarity to its nearest neighbor in the training data is reported. This procedure is repeated for 10 random shuffles of the data, and the average similarity across all shuffle and fold experiments is reported for each algorithm.

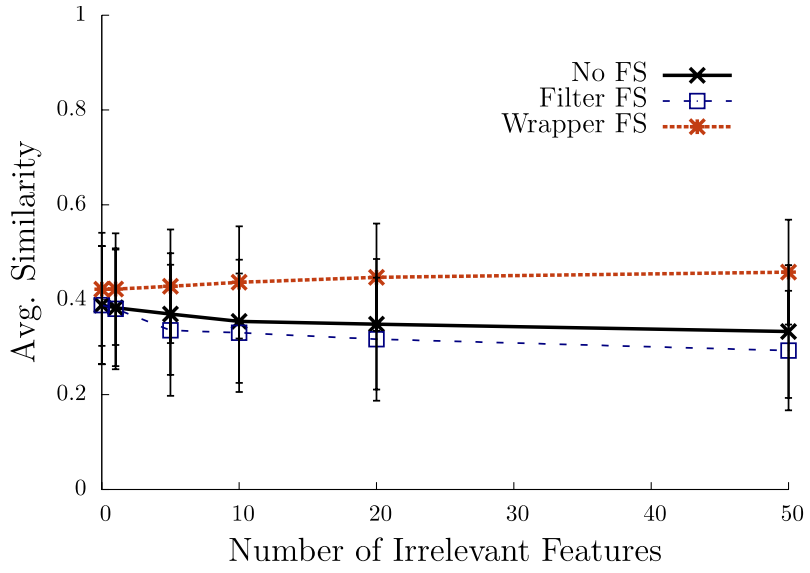

Figure 4. Average Query Similarity for Filter, Wrapper, or No Feature Selection as the Number of Irrelevant Features Increases from 0 to 50

Unlike the wrapper feature selection algorithm, the Relief-based filter feature selection algorithm does not use the similarity performance of its candidate subset to guide the search nor as a stopping criterion. This requires the filter approach to set the number of features to select beforehand. We experimented with selecting subsets of size one through seven; in general, selecting seven features gave the best performance of the algorithm and is reported in Figure 4. Additionally, the filter algorithm functions by searching the local neighborhood for cases with similar and dissimilar action plan matrices, and the number of neighbors to consult is controlled by a parameter. We experimented with one, three, and five neighbors across all scenarios of irrelevant features and numbers of selected features and found that five nearest neighbors gave slightly improved performance and is therefore the setting used in this study. It should be noted that varying the number of selected features from three to seven did not significantly alter the query results, nor did using different numbers of nearest neighbors, so the algorithm is robust to these parameters.

As shown in Figure 4, for any number of irrelevant features under study the wrapper feature selection algorithm performed the best out of the three approaches, while the filter based approach is the worst by a small margin. For all numbers of irrelevant features, each algorithm outperforms selecting a random case in response to a given query (0.28 ± 0.12). The wrapper performance shown in Figure 4 follows a surprising increasing trend as the number of irrelevant features rises from 0 to 50. As described in the previous section, the best feature subset found by the wrapper (and best overall from any subset of the seven indexing features) produced average similarity scores of around 0.42 on average, and this result is repeated for the case of 0 irrelevant features in the figure. However, as more irrelevant features are added to the data set, the retrieval similarity actually increases slightly above this value. This is due to coincidental correlations among a few of the randomly generated irrelevant features with the action plan matrices. Since the size of the case base is fixed and relatively small throughout the experiments (60 cases in total), when more irrelevant features are present, the likelihood increases that some will show false correlations with the action plan measure. This overfitting effect would likely vanish when larger case bases are used.

The diminished performance of the filter feature selection algorithm can also be explained. When no irrelevant features are present, the feature selection prefers: blue race, red state, red race, blue state, blue player name, map name, and red player name in that order, with a significant drop in relevance weights after blue state is selected. The first three features correspond to the three selected by the wrapper procedure, indicating that the Relief-based filter feature selection algorithm that we developed is effective when no irrelevant features are present. Table 4 presents the average subset precision for the filter algorithm when increasing numbers of features are present. Out of the seven features selected by the filter algorithm in each setting, an increasing number of them are irrelevant when the number of irrelevant features in the full feature set increases. In each case, at least two of the relevant selected features came from the set {blue race, red race, red state}, the optimal subset of features, however, false correlations led the selection algorithm to fill the remainder of the subset with irrelevant features, leading to the query similarity performance seen in Figure 4.

Table 4. Average Precision of Filter as the Number of Irrelevant Features Increases from 0 to 50

# Irrelevant Features:	0	1	5	10	20	50
Avg. Precision	1.0±0.0	0.86±0.0	0.54±0.10	0.36±0.13	0.25±0.12	0.15±0.10

A similar comparison of feature subset quality for the wrapper approach is given in Table 5. It is more difficult to get a sense of the general subset quality in the wrapper approach since for each fold in each shuffle of the data a different subset size can be selected, as opposed to forcing the subset set to be a fixed number as in the filter experiment. Table 5 therefore reports the average relevant subset recall, or the number of relevant features actually selected in each feature subset found by the wrapper. These numbers indicate that about 3 or 4 relevant features are selected on average, and these features include the optimal set of features found in the earlier wrapper experiments. This accounts for the consistently better performance of the wrapper algorithm in

Figure 4. In addition to these features, the wrapper tends to select a few irrelevant features in each subset, giving rise to the slight improvement in query similarity noted in Figure 4. On a larger case base, the irrelevant features would be less likely to exhibit such correlations and thus be less likely to be selected by the wrapper procedure.

Table 5. Average Recall of Wrapper as the Number of Irrelevant Features Increases from 0 to 50

# Irrelevant Features:	0	1	5	10	20	50
Avg. Recall	0.56±0.22	0.56±0.22	0.49±0.22	0.46±0.25	0.45±0.22	0.41±0.24

4.4. Case Clustering

The final part of experimental results is concerned with the ability to cluster cases based on their action plan matrices. High quality clustering results would enable a single value (the cluster ID) to replace the action plan matrix as the class label for each case in the data set. This change would allow the full contingent of supervised filter feature selection algorithms to be tested on the case base data without the need for special engineering (as in the Relief algorithm described above). To this end, we experimented with an implementation of an agglomerative hierarchical clustering algorithm to cluster cases based on action plan matrices using the proposed general similarity measure described in Section 3.2.1.

There are several parameters to set in an agglomerative clustering algorithm, such as which cluster member to use to measure distance between clusters (Min, Max, or Average linkage), and when should the agglomerative process terminate (at some specified number of clusters, or once the nearest remaining clusters exceed some distance threshold). Due to time constraints on the project, we did not perform a rigorous test across the parameter space, but we did try several variations and report clustering result summary statistics in Table 6.

Table 6. Summary Statistics of Clusters Found by Hierarchical Clustering Using Min Linkage

Number of Clusters	Average Cluster Cohesion	Average Cluster Separation
4	0.46±0.03	0.32±0.05

These results report findings based on four clusters, capturing 40 out of 60 cases. The remaining 20 cases were not merged into clusters with these parameter settings. The four clusters all had high cohesion scores, indicating internally similar action plans, while similarity between clusters was lower, at around 0.32. These trends indicate that these clusters might be located somewhat near each other, though with some specific structural differences. It is worth pointing out that when the 20 singleton clusters are added to the results the separation between groups decreases to 0.21; a lower similarity score than the average similarity between randomly chosen cases. Subjective validation is needed at this stage to assess whether or not the cluster members are

truly similar to one another, either by manual inspection of the replay data that generated the cases, or automatic execution of the strategy information encoded in the cases. Given the short duration of this project, we were unable to investigate these results further at this time, but note that the preliminary results do suggest that clustering of cases is feasible and can provide meaningful information. Furthermore, any improvements made to the general similarity measure will also positively impact the clustering results. Once high-quality clustering results are achieved and verified, the cluster labels can simply be used in supervised learning experiments to continue feature selection algorithm improvements for case based planning.

4.5. Discussion

One key thing to notice from Table 2 is that the average similarity between any case and its nearest neighbor (excluding itself) is only around 0.51. If the case base contains similar cases, as would be reasonable to expect, then this result seems low. This can be explained from two perspectives: either there were too few cases in the sample case base, or the similarity measure used is too rigid to fully capture case similarity. The first possibility may be easily explored by working with a case base engineered to include several clusters of cases, where each cluster has been subjectively observed to contain similar cases. The rigidity of the proposed similarity measure is in respect to the measuring of column vectors against one another. If we consider two cases which were identical except for an initial execution delay, then the current similarity measure will give a poor similarity score despite the two plans being different only with respect to the step in which they were initiated. Possible future extensions to the measure can include forms of dynamic time warping which can account for such temporal variations and make the proposed measure more robust.

From the wrapper experiments we can conclude that feature selection shows promise in enabling more compact indexing structures that can outperform the full set. We can also see that the set of static features does not capture the spread of information that is contained in the action plan since there are more similar cases for given queries than can be found using a nearest neighbor search in any subset of the seven current features. We could view the nearest neighbor retrieval as a classification algorithm. In this sense, other classification algorithms could be used to retrieve more similar cases and may be more successful (we see there is room for improvement by looking at the 0.42 results from exhaustive and 0.51 from optimal).

Filter feature selection experiments demonstrated that feature selection for CBP could be conducted without performing expensive querying to measure the goodness of a feature subset, a necessary capability when considering large case bases with many features. The experiments conducted along this line also pointed out two important areas to consider when pursuing future research. First, the set of indexing features must be related to the class label. Second, appropriate filter feature selection techniques must be chosen depending on the data composition of the available feature set. Since a filter feature selection algorithm relies on the intrinsic relationship between features and a class label (or the action plan matrix in this work), there must be a strong predictive relationship between these two entities in order for a filter strategy to work. In these experiments, we have learned that several of the currently used indexing features yield little

information with respect to the action plan matrix, leading the feature selection algorithm to prefer irrelevant features. If the action plan similarity measure makes sense for CBP, then it appears that the set of indexing features should be augmented to include other features which relate more directly to the action plan matrix. In any case, an appropriate algorithm must be selected based on the characteristics of the data, or else the feature selection results could be harmed. In this short-duration project, we explored the Relief algorithm, which relies on measuring nearest neighbor distance between samples in the feature space. In this study, we measure distance between cases in the case base along a subset of *categorical* indexing features, which limits the type of distance metrics that could be used. Other filter evaluation algorithms could be tested and may work better than Relief for case base data; however, the chief concern is still including features in the full subset that are relevant to the chosen class label.

5. Conclusions

In this project, we have addressed two main challenges in feature selection for CBP: evaluating the relevance of a feature subset and validating the result of a feature selection algorithm for CBP. The proposed solutions to these challenges are centered around an innovative idea which treats the plan portion of a case as its supervised label, and represents each label in the form of a sparse matrix, enabling the design of a general similarity measure to measure the match between two cases based on their plan portions. Based on the general similarity measure, we have developed several wrapper and filter methods for CBP. In particular, we have developed wrapper methods which use the aggregated leave-one-out loss of the training set as the subset evaluation measure. In addition, we have developed two filter approaches which either directly or indirectly use the general similarity measure to evaluate feature relevance. The indirect approach is characterized by a novel preprocessing procedure which first clusters the plan portions of all cases in a case base into a number of categories based on the general similarity measure, and then converts the original case base into a case base with categorical case labels. Based on the general similarity measure, we have also developed an N-fold cross validation (CV) procedure for result validation.

The experimental results have shown that the proposed feature selection algorithms are capable of identifying relevant features among a large number of candidate features for indexing the cases. The proposed case clustering approach is promising in discovering high-level strategies (clusters) among the case base and thus enabling the application of state-of-the-art supervised feature selection algorithms on the case base. The automated feature selection capability provided by this project enables domain experts to include a sufficiently rich set of state features in the case base without the burden of manually deciding on an optimal set of features. Future effort will enable feature selection to adaptively include new features based on the dynamics of the battle space and automatically identify features from past experiences to construct the case base. Several immediate extensions of the effort of this project include: (1) improving the general similarity measure for plans by accounting for temporal variations of the same plan and studying the effect of variations of the general measure on the feature selection results; (2) an in-depth study of the effectiveness of case clustering for identifying high-level strategies based on case bases with known strategies; (3) developing unsupervised feature selection methods which identify relevant features directly from the plan portions of cases (the plan matrices) – features

that are relevant to CBP should contribute to the clustering of the plans into different strategies; and (4) studying the effectiveness of feature selection algorithms based on enriched sets of features (provided by the DEEP team or identified by unsupervised feature selection).

The proposed CV procedure has also been shown useful for result validation of feature selection for CBP. The procedure is broadly applicable for evaluating alternative strategies of CBP, for example, using different similarity measures for nearest neighbor search in plan retrieval or different mechanisms for aggregating the plans of several nearest neighbors. Nevertheless, the use of the general similarity measure in the CV procedure has its own limitations compared to real-time execution of retrieved plans. The performance of a strategy under evaluation may be sensitive to variations in the specific measure used, and the best plan according to the measure may not always be the best plan in execution. A future research direction is to put the top performers of our proposed algorithms in action, and cross check the performance of these algorithms by real-time plan execution based on the current and improved versions of attack managers.

6. Appendix

The content in this appendix reports additional research findings during the no-cost-extension period from August 1 to September 30, 2012. We present an improved similarity measure for plans by accounting for temporal variations of the same plan, and report results of wrapper and filter feature selection based on the new similarity measure. We also compare the new results of feature selection to those from the previous similarity measure reported in Section 4.

6.1. Improved Similarity Measure

One issue about using the general loss function in Equation (1) to measure case similarity is the misalignment of actions in time between two cases of same strategy (build order). Players take varying amounts of time to execute their strategies. Similar strategies in this sense would be trivial to spot by a human, but difficult to detect by the current measure. If the same actions in two cases of same strategy are even one time step off, the current similarity measure will consider the two cases dissimilar. One way to overcome this limitation is to apply time warping to the action matrices. The idea is to take into account units from a number of steps earlier and later for a given time step. A discount factor is applied depending on how many time steps apart from the time step in question. Specifically, the time warped value on matrix **P** for unit *u* at time step *t* by considering *k* time steps before and after *t* is defined as:

$$p_{u,t} = \sum_{1 \le l \le k} r(l) p_{u,t-l} + p_{u,t} + \sum_{1 \le l \le k} r(l) p_{u,t+l} \,, \tag{4}$$

where the discount factor is given by

$$r(l) = 1 - \frac{l-1}{k} \,.$$

The parameter *k* controls the tradeoff between accommodation of time-misaligned data and the preservation of build order. In our experiments we set *k* equal to 3, considering the value of 3

corresponds to roughly one minute delay in taking actions. Experimental results show that this setting is able to correct the time misalignment for most cases with slight time differences in executing the same strategy. After applying time warping conversion of the case base, we apply the same general loss function in Equation (1) in measuring the similarity between two cases.

Table 7 below shows the results of average leave-one-out cross validation (LOOCV) performance of the improved similarity measure in comparison with the previous version without time warping. To compute the LOOCV performance, for each left-out case (query case), its similarity to its nearest neighbor, farthest neighbors, and its average similarity to all other cases in the case base is calculated, and the performance is averaged over all cases. Since the action plan matrices are normalized and unit weights are used on all columns, the values of the similarity measure are bounded between 1 (most similar) and 0 (least similar) in all experiments. Based on the previous version of the similarity measure, the nearest neighbor similarity to a query case scores on average around midway in the possible range of [0,1], which is somewhat lower than could be expected. The furthest neighbor similarity is near 0.0. Based on the new similarity measure, the values in all three columns are higher. Most importantly, the average nearest neighbor similarity has improved significantly from around midway of the possible range to being very close to 1, while the average furthest neighbor similarity remains close to 0 despite a slight increase. Such results verify that the time warping mechanism effectively improves the general similarity measure.

Table 7. LOOCV Performance under the Previous and New Similarity Measures

	Nearest Neighbor	Furthest Neighbor	Average Neighbor
Previous Similarity Measure	0.51±0.10	0.04±0.03	0.28±0.07
New Similarity Measure	0.97±0.02	0.18±0.05	0.65±0.04

When comparing the results, it is worth noting that the two similarity measures use different forms of distance function $d(\mathbf{p}_t, \mathbf{p}_t)$ for Equation (1). The previous similarity measure uses l_1 norm which works better than Cosine distance, while the new similarity measure uses Cosine distance which gives the best similarity results. The superior performance of Cosine distance in the new experiments is that the action matrices become more densely populated with nonzero values due to the time warping operation. In the next two sections, we report wrapper and filter feature selection results based on the new similarity measure and compare the new results to those from the previous similarity measure. In order to make the two sets of results directly comparable, we rerun the experiments reported in Sections 4.2 and 4.3 using Cosine distance as the base distance function for the previous similarity measure.

6.2. Wrapper Feature Selection

Table 8 reports the average value and standard deviation on the LOOCV similarity for the full feature set and the best feature subset found by the wrapper process under both the new similarity

measure and the previous similarity measure (with the base distance function changed to Cosine distance).

Table 8. LOOCV Similarity under the Previous and the New Measures for the Full Feature Set and Selected Subset by the Wrapper

	Full Feature Set	Wrapper Selected Feature Subset
Previous Similarity Measure	0.11±0.05	0.12±0.07
New Similarity Measure	0.17±0.11	0.19±0.11

The first notable observation is that the similarity values for the previous similarity measure using Cosine distance as the base distance function are significantly lower than those reported in Table 3. Second, the values are higher based on the new similarity measure, which is expected since the time warping performed on the action matrices before similarity computation can correct possible misalignment of actions in time between two similar cases.

When comparing the results of wrapper feature selection to the LOOCV similarity among action matrices based on the new similarity measure (the second row of Table 7), we can see that the similarity of two cases which are deemed as nearest neighbors based on the selected feature subset in the state space are actually far less than the average neighbor similarity among all action matrices. This observation appears to contradict with the observation made in Section 4.2 that when using the subset identified by the wrapper, the action similarity between a query and its nearest neighbor (in the state space) is greater than that of itself to all other cases (the Average Neighbor similarity). In fact, this is mainly due to the limitation of the previous similarity measure which does not properly handle cases of the same or similar strategy with misaligned actions in time. The new similarity measure addresses this limitation and results in much higher similarity values for Nearest Neighbor and Average Neighbor, and therefore, allows a more realistic baseline to evaluate the performance of feature selection. We did note in Section 4.2 that the similarity of the nearest neighbor based on the full feature set or the selected feature subset is significantly less than the most similar case to the query that exists in the case base. This finding indicates that there is more information in the action plan matrix than is currently being captured in the feature subset. The new results are consistent with our previous findings in this regard.

Overall, we conclude that the features that are currently used to index the case base are far from enough to capture the similarity of cases in terms of actions. Additional features selected by an unsupervised approach that directly works on the action matrices can help to enrich the set of indexing features, in addition to features hand-picked based on domain knowledge.

6.3. Filter Feature Selection

The setup of experiments for this section is the same as the one used in Section 4.3 except that the base distance function used for the previous similarity measure is changed to Cosine distance. Figure 5 reports the results of average LOOCV similarity for wrapper and filter feature selection

with the previous and the new similarity measures. The reference curve of no feature selection based on the previous similarity measure is also included. We can observe similar trends as those from Figure 4 in Section 4.3. Wrapper feature selection performed the best out of the three methods, while the filter method is the worst among the three. For both the wrapper and filter methods, the similarity values under the new measure are in general higher than those under the previous measure. For the wrapper method, the trend of increasing performance as the number of irrelevant feature rises is more pronounced in Figure 5 than the subtle trend shown in Figure 4. As explained in Section 4.3, this uprising trend is due to coincidental correlations among a few of the randomly generated irrelevant features with the action plan matrices. As the number of irrelevant features increases, the best subset selected is more likely to contain irrelevant features that are superficially correlated with the similarity of the action matrices. Since the upper bound on the similarity values has improved from around 0.5 to 1 under the new similarity measure, it leaves more room for the wrapper method to demonstrate the artifact of increasing similarity with more irrelevant features.

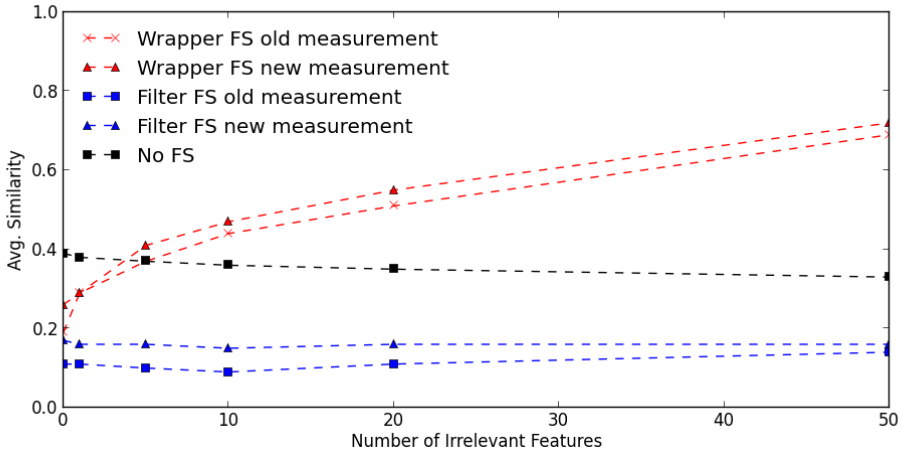

Figure 5. Average Query Similarity under the Previous and the New Measures for Filter, Wrapper, or No Feature Selection as the Number of Irrelevant Features Increases from 0 to 50

Table 9 presents the average subset precision for the filter algorithm when increasing numbers of features are present. Similar trends can be observed as in Table 4 from Section 4.3. Out of the seven features selected by the filter algorithm in each setting, an increasing number of them are irrelevant when the number of irrelevant features in the full feature set increases. In each case, at least two of the relevant selected features came from the set {blue race, red race, red state}, the optimal subset of features, however, false correlations led the selection algorithm to fill the remainder of the subset with irrelevant features, leading to the poor query similarity performance seen in Figure 5. The difference between the previous and the new measure is insignificant.

Table 9. Average Precision of Filter under the Previous and the New Measure as the Number of Irrelevant Features Increases from 0 to 50

# Irrelevant Features:	0	1	5	10	20	50

Previous Measure	1.0±0.0	0.86±0.0	0.54±0.10	0.36±0.13	0.25±0.12	0.15±0.10
New Measure	1.0±0.0	0.86±0.0	0.57±0.10	0.40±0.14	0.26±0.14	0.16±0.11

7. Bibliography

[1] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[2] J. A. Carozzoni, J. H. Lawton, C. DeStefano, A. J. Ford, J. W. Hudack, K. K. Lachevet, and G. R. Staskevich. Distributed episodic exploratory planning (DEEP). AFRL-RI-RS-TR-2008-279. Technical report, Air Force Research Laboratory, 2008.

[3] W. Dahm. Technology horizons: A vision for air force science & technology during 2010-2030. Technical report, Air Force, 2010.

[4] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artifficial Intelligence Research*, 13:227–303, 2000.

[5] J. G. Dy and C. E. Brodley. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845–889, 2004.

[6] J. G. Dy, C. E. Brodley, A. C. Kak, L. S. Broderick, and A. M. Aisen. Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):373–378, 2003.

[7] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[8] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[9] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292, 1996.

[10] S. M. Lavelle. *Planning Algorithms*. Cambridge University Press, 2006.

[11] Yu Li and B. L. Lu. Feature selection based on loss-margin of nearest neighbor classification. *Pattern Recognition*, 42(9):1914–1921, 2009.

[12] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Boston: Kluwer Academic Publishers, 1998.

[13] H. Liu and R. Setiono. Feature selection and classification - a probabilistic wrapper pproach. In T. Tanaka, S. Ohsuga, and M. Ali, editors, *Proceedings of the Ninth International Conference on Industrial and Engineering Applications of AI and ES*, pages 419–424, Fukuoka, Japan, 1996.

[14] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 319–327, 1996.

[15] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(4):491–502, 2005.

[16] K. Mishra, S. Ontanon, and A. Ram. Situation assessment for plan retrieval in real-time strategy games. In *Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR 2008)*, pages 355–369, 2008.

[17] A. Navot, L. Shpigelman, N. Tishby, and E. Vaadia. Nearest neighbor based feature selection for regression and its application to neural activity. In *Proceedings of Neural Information Processing Systems Conference*, pages 995–1002, 2006.

[18] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27:1226–1238, 2005.

[19] M. Robnik-Sikonja and I. Kononenko. An adaptation of relief for attribute estimation in regression. In *Proceedings of Fourteenth International Conference on Machine Learning*, pages 296–304, 1997.

[20] R. Wright, S. Loscalzo, and L. Yu. Embedded incremental feature selection for reinforcement learning. In *the 3rd International Conference on Agents and Artificial Intelligence (ICAART- 2011)*, 2011.

[21] L. Yu and H. Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *Proceedings of the twentieth International Conference on Machine Learning (ICML)*, pages 856–863, 2003.

[22] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research (JMLR)*, 5:1205–1224, 2004.

[23] L. Yu and H. Liu. Redundancy based feature selection for microarray data. In *Proceedings of the Tenth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 737–742, 2004.

Symbols, Abbreviations and Acronyms

CBP – Case Based Planning
CV – Cross Validation
DEEP – Distributed Episodic Exploratory Planning
EBAR – Experience Based Adaptive Replanning
FCBF – Fast Correlation Based Filter
LOOCV – Leave One Out Cross Validation
MDP – Markov Decision Process
MRMR – Minimum Redundancy Maximum Relevancy